$$\left(\exists i \in [0, dima).\ \left(\exists j \in [0, dimb).\ a[i] = b[j]\right)\right)$$

```
int member (int el, int a[], int dim)
{ int i = 0;
  int trovato = 0;
  while (i < dim && !trovato)
      if (el == a[i]) trovato = 1;
      else  i++;
  return trovato;
}

int esiste (int a[], int dima,
            int b[], int dimb)
{ int i = 0;
  int trovato = 0;
  while (i < dima && !trovato)
      if (member(a[i], b, dimb)) trovato = 1;
      else  i++;
```

```
    return trovato;
}
```

Tutti gli elementi di a compaiono anche in b!

$$\left(\forall i \in [\emptyset, \mathrm{dim}\, a).\ \left(\exists j \in [\emptyset, \mathrm{dim}\, b).\ a[i] = b[j]\right)\right)$$

$$\left(p(i)\right)$$

dominio di i

proprietà (formula logica che coinvolge i)

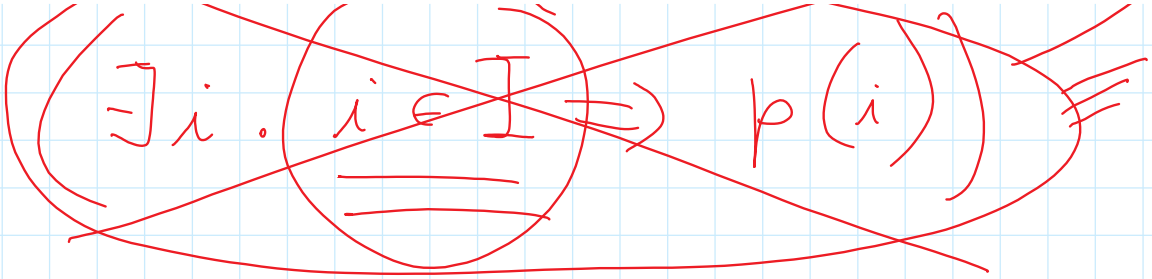
abbreviazione di

$$\left(\forall i.\ i \in I \Rightarrow p(i)\right)$$

$$\left(\exists i \in I.\ p(i)\right)$$

abbreviazione di

$$\left(\exists i.\ i \in I \wedge p(i)\right)$$



$$(\forall i \in [0, dima) . (\exists j \in [0, dimb) . a[i] = b[j]))$$

$$(\forall i . i \in [0, dima) \Longrightarrow (\exists j . j \in [0, dimb) \wedge a[i] = b[j]))$$

```
int member (int el, int a[], int dim)
{ ...
}

int pertutt (int a[], int dima,
             int b[], int dimb)
{ int i = 0;
  int ok = 1;
  while (i < dima && ok)
    if (! member (a[i], b, dimb) ok = 0;
    else i++;
```

return OK;
}

$$(\forall i \in I \,.\, p(i)) \equiv (\forall i \,.\, i \in I \Rightarrow p(i))$$

$$(\forall i \in \{\} \,.\, p(i)) \equiv (\forall i \,.\, i \in \{\} \Rightarrow p(i))$$

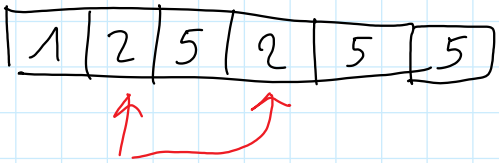
$\emptyset$ — Vera

$$(\exists i \in I \,.\, p(i)) \equiv (\exists i \,.\, i \in I \wedge p(i))$$

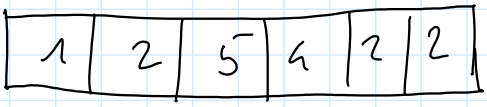
$$(\exists i \in \{\} \,.\, p(i)) \equiv (\exists i \,.\, i \in \{\} \wedge p(i))$$

falsa

Dato un array vogliamo verificare che esiste un elemento che compare esattamente due volte



Vera



falsa

Cardinalità di un insieme finito

$\#I \qquad |I|$

è il numero degli elementi

$\#\{1,5,4\} = 3$

$\#\{1,4,5,\cancel{1}\} = 3$

$$\left(\exists i \in [0, dim).\ \left(\{j \mid j \in [0, dim) \wedge a[i] = a[j]\} = 2\right)\right)$$

$(\exists i : i \in [\emptyset, dim) \cdot (\#\{j \mid j \in [\emptyset, dim) \wedge a[i] = a[j]\} = 2))$

```
int conta (int el, int a[], int dim)
{ int i;
  int num = 0;
  for (i = 0; i < dim; i++)
      if (a[i] == el) num++;
  return num;
}

int esattamentedue (int a[], int dim)
{ int i = 0;
  int trovato = 0;
  while (i < dim && !trovato)
     if (conta (a[i], a, dim) == 2)
                    trovato = 1;
     else i++;
  return trovato;
```

```
} return trovato;
```

-1	3	5	5	1	3

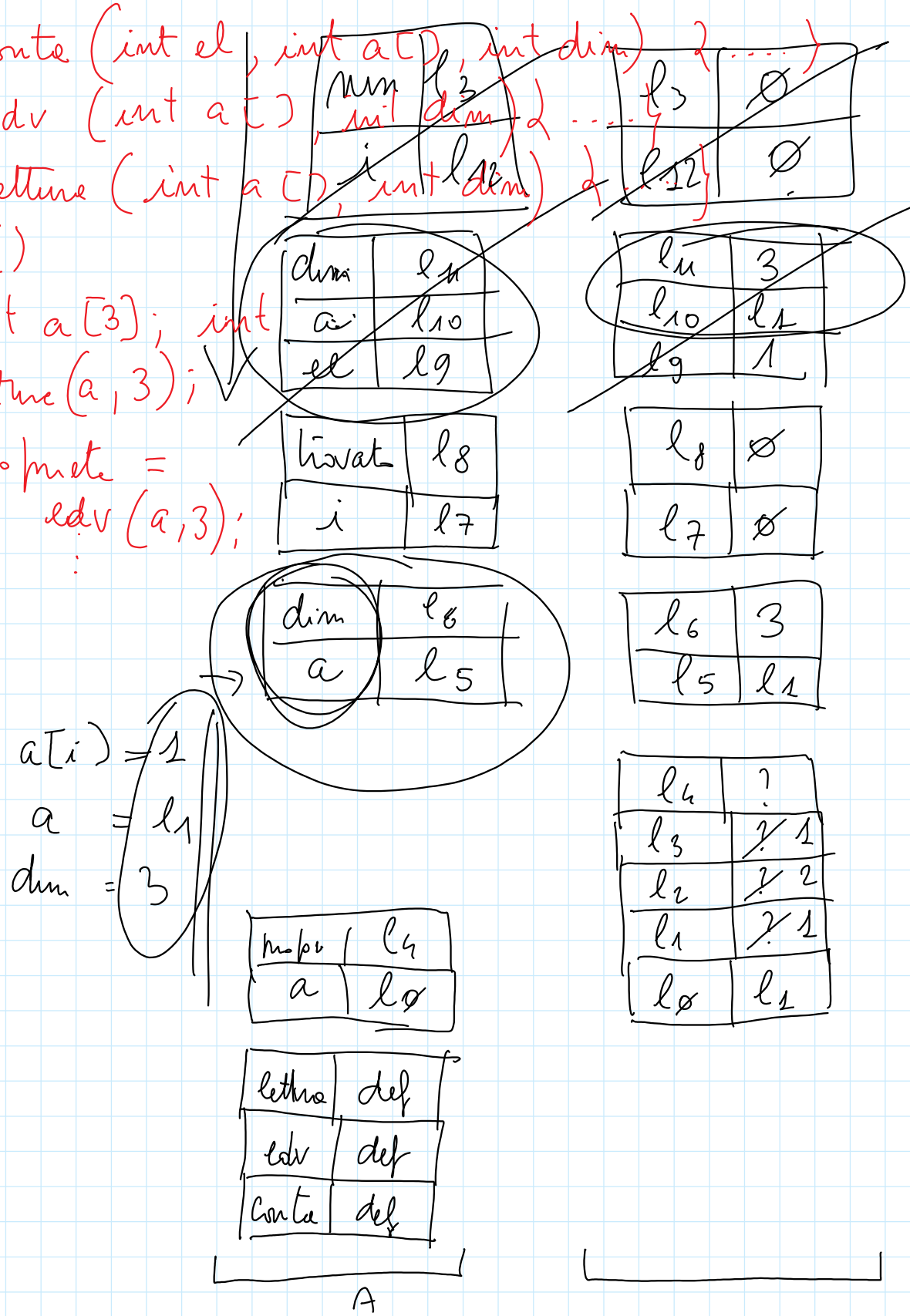
Vera

$$\left(\forall i \in [0, dim).\ \left(\#\{j \mid j \in [0, dim) \wedge a[i] = a[j]\} = 2\right)\right)$$

```
int conta (int el, int a[], int dim)
{ ... }

int tuttidue volte (int a[], int dim)
{ int i = 0;
  int ok = 1;
  while (i < dim && ok)
    if (conta(a[i], a, dim) != 2) ok = 0;
    else i++;
  return ok;
}
```

mercoledì 3 ottobre 2018 12:18

```
#include <stdio.h>
int conta (int el, int a[], int dim) {....}
int edv (int a[], int dim) {....}
void lettura (int a[], int dim) {....}
main()
{ int a[3]; int
  lettura(a,3);
  risposta =
      edv(a,3);
      :
```



Cercare il valore massimo in un array.

a | -3 | 2 | 5 | -20 | 10 | ⟶ (10)

```
int max (int a[], int dim)
{ int i;
  int vmax = a[0];
  for (i = 1; i < dim; i++)
      if (a[i] > vmax) vmax = a[i];
  return vmax;
}

main()
{ int a[100];
  int v;
  lettura (a, 100);
  v = max (a, 100);
  :
```

Se voglio, oltre al valore massimo, il suo indice?

a

2	3	-5	7	-4	-2
0	1	2	3	4	5

Valore massimo 7

indice 3

```
main()
{  int a[100];
   int v;
   int iv;
   v = max(a, 100, &iv);
}
```

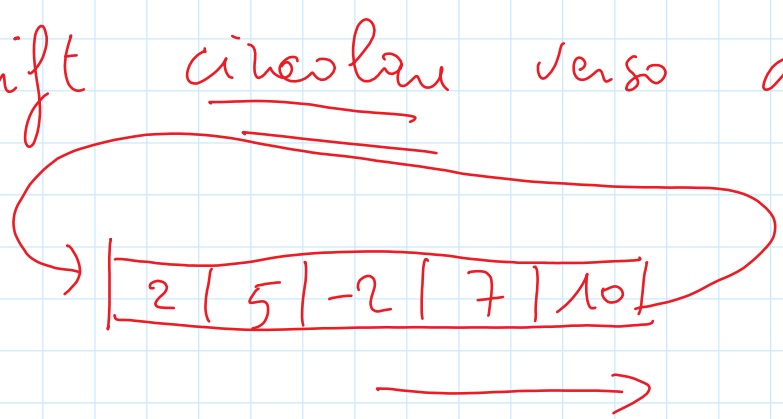
```
int max(int a[], int dim, int* indice)
{  int i;
   int vmax = a[0];
   *indice = 0;
   for(i=1; i<dim; i++)
```

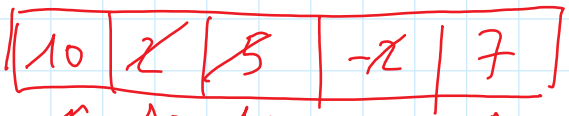
```
if (a[i] > vmax)
    {
    vmax = a[i];
    *indice = i;
    }

return vmax;
}
```

```
main()
{  int a[100];
   int v;
   int iv;
   lettura(a, 100);
   max(a, 100, &v, &iv);
        :
```

```
void max(int a[], int dim,
         int *vmax, int *indice)
{  int i;
   *vmax = a[0];
   *indice = 0;
   for(i=1; i<dim; i++)
      if(a[i] > *vmax)
         {  *vmax = a[i];
            *indice = i;
         }
}
```

Shift circolare verso destra





procedura

```
void shift (int a[], int dim)
{ int i;
  for (i=0; i<dim-1; i++)
      a[i+1] = a[i];
```

```
void shift (int a[], int dim)
{ int temp = a[dim-1];
  int i;
  for(i = dim-1; i>0; i--)
      a[i] = a[i-1];
  a[0] = temp;
}
```

temp = 2